Marcelo Augusto Stefanini Faria

**Sheet Gen**

Software para criação de partituras – Relatório TCC I

Limeira

2019

Marcelo Augusto Stefanini Faria

**Sheet Gen**

Software para criação de partituras - Relatório TCC I

Trabalho de Conclusão de Curso I apresentado à Faculdade de Tecnologia da UNICAMP, para obtenção do título de Bacharel em Sistemas de Informação, sob a orientação do Prof. Dr. João Roberto Bertini Junior.

Limeira

2019

Resumo - Sheet Gen

Este trabalho tem como objetivo resolver de um dos grandes problemas no universo da música: a escrita de partituras.

Existem diversos softwares para escrever partituras, porém além de se apresentarem muito difíceis de usar, requerirem muita experiência e prática para que se escreva rápido, ainda são muito lentos.

A idealização desse software se dá justamente na tentativa de vencer essas grandes barreiras utilizando as mais altas tecnologias disponíveis no mercado de informática.

A escrita das partituras será feita a partir da execução de um instrumentista. O sistema, através de diversas análises, entende o que está sendo executado e retorna o produto final: a partitura.

[1. Versão do Documento 6](#_Toc11768666)

[2. Introdução 6](#_Toc11768667)

[3. Linguagem de programação 6](#_Toc11768668)

[4. Bibliotecas a utilizar 6](#_Toc11768669)

[5. Algoritmos a utilizar – Must Have 6](#_Toc11768670)

[5.1. Funcionalidades Básicas 6](#_Toc11768671)

[5.1.1. Captar áudio pelo microfone 6](#_Toc11768672)

[5.1.2. Plotar das ondas sonoras em gráfico 6](#_Toc11768673)

[5.1.3. Implementar a taxa de atualização do gráfico de ondas sonoras ( Quanto maior , mais custo computacional ) 7](#_Toc11768674)

[5.1.4. Retornar valor em Hertz (Hz) de uma posição do gráfico 7](#_Toc11768675)

[5.1.5. Identificar picos do gráfico de ondas sonoras 7](#_Toc11768676)

[5.1.6. Retornar valor da frequência em Hertz (Hz) do pico identificado 7](#_Toc11768677)

[5.1.7. Identificar mudança de pico 7](#_Toc11768678)

[5.1.8. Retornar valor em milisegundos ou segundos da mudança de um pico para o outro 7](#_Toc11768679)

[5.2. Funcionalidades Intermediárias 7](#_Toc11768680)

[5.2.1. Implementar sistema de configuração inicial 7](#_Toc11768681)

[5.2.2. Normalizar ondas sonoras no que se diz respeito à intensidade do som 7](#_Toc11768682)

[5.2.3. Criar templates das páginas, onde serão inseridos os pentagramas, utilizando alguma biblioteca de plotagem 7](#_Toc11768683)

[5.2.4. Criar as figuras musicais através de bibliotecas de desenho para serem inseridas no template dos pentagramas 7](#_Toc11768684)

[5.2.5. Se necessário, integrar bibliotecas de plotagem e de desenho para gerar a partitura final 7](#_Toc11768685)

[5.2.6. Relacionar os conceitos musicais com as saídas das funcionalidades básicas 7](#_Toc11768686)

[5.2.7. Converter a partitura gerada para .pdf 7](#_Toc11768687)

[5.3. Funcionalidades Avançadas 7](#_Toc11768688)

[5.3.1. Gerar a partitura em tempo real 7](#_Toc11768689)

[6. Algoritmos a utilizar – Nice to Have 7](#_Toc11768690)

[6.1. Funcionalidades Gerais 7](#_Toc11768691)

[6.1.1. Implementar animação de um “cursor” indicando o progresso da música 7](#_Toc11768692)

[6.1.2. Implementar animação para que a plotagem dos desenhos seja fluida e bonita 7](#_Toc11768693)

[6.1.3. Identificar várias melodias ao mesmo tempo ( mais de uma voz ) 7](#_Toc11768694)

[6.1.4. Animação de load equanto o sistema gera o .pdf da partitura 8](#_Toc11768695)

[6.1.5. Possibilitar o usuário fazer correções na partitura gerada 8](#_Toc11768696)

[6.1.6. Gerenciar as informações da partitura e metadados 8](#_Toc11768697)

[6.1.7. Possibilitar a inserção de letra na partitura 8](#_Toc11768698)

[6.1.8. Implementar conceitos de Machine Learning para que, a cada vez que o sistema for usado, melhore sua precisão ao gerar as partituras 8](#_Toc11768699)

[6.1.9. Implementar sistema de aviso que detecta se o local está apropriado ou não para gerar partituras 8](#_Toc11768700)

[6.1.10. Implementar sistema capaz de reconhecer o andamento & bpm, fórmula de compasso e armadura de clave 8](#_Toc11768701)

[7. Tarefas importantes 8](#_Toc11768702)

[7.1. Categorizar timbres 8](#_Toc11768703)

[7.2. Definir um instrumento ideal para o desenvolvimento 8](#_Toc11768704)

[7.3. Organizar a exibição da interface gráfica em toda sua execução 8](#_Toc11768705)

[7.3.1. Tela inicial 8](#_Toc11768706)

[7.3.2. Tela de execução de uma seção 8](#_Toc11768707)

[7.3.3. Tela de encerramento de uma seção 8](#_Toc11768708)

[8. Conclusão 8](#_Toc11768709)

Sheet Gen

# Versão do Documento

* Versão 1.2

# Introdução

O sistema de computadores Sheet Gen é capaz de imprimir partituras através da captação de sons pelo microfone do computador. Isso é possível graças a diversos procedimentos de análise de dados que, trabalhando juntos, conseguem retornar exatamente o que foi executado pelo instrumentista.

Utilizando Inteligência Artificial, Aprendizado de Máquina e diversos outros algoritmos, Sheet Gen consegue entregar partituras com alta confiabilidade e precisão além de alta performance em escrevê-las em tempo real enquanto o usuário utiliza-o.

# Linguagem de programação

* Python (versão ainda não foi decidida)

# Bibliotecas a utilizar

* pyaudio
* numpy
* matplotlib.pyplot
* pandas
* plotly
* pydot
* turtle.py
* scikit-learn

# Algoritmos a utilizar – Must Have

## Funcionalidades Básicas

### Captar áudio pelo microfone

Uma das premissas básicas do sistema é que o produto final que é a partitura seja gerada a partir da música que um instrumentista executar. A melhor forma de se fazer isso é utilizando um microfone de boa qualidade e em um ambiente calmo, sem barulho externo.

### Plotar das ondas sonoras em gráfico

Plotar o gráfico das ondas sonoras captadas pelo microfone é um passo importante para a execução desse trabalho, pois com ele será possível identificar as notas através de sua frequência em Hertz (Hz)

Identificar melhor formato para exibição de ondas

### Implementar a taxa de atualização do gráfico de ondas sonoras

Um dos grandes desafios desse item é que quanto maior a taxa de atualização, mais nítido, fluido e legível fica o gráfico, porém maior o custo computacional. Então, implementar a possibilidade do usuário mudar a taxa de atualização seria interessante para que o sistema possa ser rodado em computadores mais simples. Neste caso, é muito importante atribuir um limite de segurança para evitar quebras no sistema.

Identificar melhor taxa para atualização das ondas

### Retornar valor em Hertz (Hz) de uma posição do gráfico

### Identificar picos do gráfico de ondas sonoras

### Retornar valor da frequência em Hertz (Hz) do pico identificado

### Identificar mudança de pico

### Retornar valor em milisegundos ou segundos da mudança de um pico para o outro

## Funcionalidades Intermediárias

### Implementar sistema de configuração inicial

### Normalizar ondas sonoras no que se diz respeito à intensidade do som

### Criar templates das páginas, onde serão inseridos os pentagramas, utilizando alguma biblioteca de plotagem

### Criar as figuras musicais através de bibliotecas de desenho para serem inseridas no template dos pentagramas

### Se necessário, integrar bibliotecas de plotagem e de desenho para gerar a partitura final

### Relacionar os conceitos musicais com as saídas das funcionalidades básicas

Retornar as conclusões e imediatamente desenhar o que foi retornado na partitura.

### Converter a partitura gerada para .pdf

## Funcionalidades Avançadas

### Gerar a partitura em tempo real

Otimizar ao máximo para que seja o mais instantâneo possível

# Algoritmos a utilizar – Nice to Have

## Funcionalidades Gerais

### Implementar animação de um “cursor” indicando o progresso da música

### Implementar animação para que a plotagem dos desenhos seja fluida e bonita

### Identificar várias melodias ao mesmo tempo ( mais de uma voz )

* + Identificar vários picos do gráfico de ondas sonoras
  + Retornar valor da frequência em Hertz (Hz) dos picos identificados
  + Identificar várias mudanças de picos
  + Retornar valor em milisegundos ou segundos das mudanças de vários picos

### Animação de load equanto o sistema gera o .pdf da partitura

### Possibilitar o usuário fazer correções na partitura gerada

### Gerenciar as informações da partitura e metadados

* + Nome do Arquivo
  + Título da partitura
    - Subtítulo
  + Autor
  + Direitos autorais
  + Descrição

### Possibilitar a inserção de letra na partitura

### Implementar conceitos de Machine Learning para que, a cada vez que o sistema for usado, melhore sua precisão ao gerar as partituras

* + Treinar o sistema com algumas situações mais complexas que ele pode se deparar
  + Treinar o sistema com situações em que esteja gerando a partitura de forma incorreta
  + Gerar uma porcentagem de 0% a 100% de confiança para cada nota e, ao final da execução, ressaltar os pontos mais críticos.
  + Guardar na memória tudo o que foi executado para que, depois que o usuário parar de executar, verificar se existe alguma melhoria que pode ser feita através do que o computador já aprendeu

### Implementar sistema de aviso que detecta se o local está apropriado ou não para gerar partituras

### Implementar sistema capaz de reconhecer o andamento & bpm, fórmula de compasso e armadura de clave

# Tarefas importantes

## Categorizar timbres

## Definir um instrumento ideal para o desenvolvimento

## Organizar a exibição da interface gráfica em toda sua execução

### Tela inicial

### Tela de execução de uma seção

### Tela de encerramento de uma seção

# Conclusão